AD-A086 519 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 15/7
GROUND MOVEMENT MODELLING IN THE STAR COMBAT MODEL.(U)
MAY 80 J K HARTMAN MIPR-32CAAM01
UNCLASSIFIED NPS55-80-021 NL

END
DATE
FILMED
8-80
DTIC

NPS55-80-021

NAVAL POSTGRADUATE SCHOOL

Monterey, California

GROUND MOVEMENT MODELLING
IN THE
STAR COMBAT MODEL

by

James K. Hartman

May 1980

Approved for public release; distribution unlimited.

Prepared for:
The U.S. Army Training & Doctrine Command
Fort Monroe, VA.

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral J. J. Ekelund
Superintendent

Jack R. Borsting
Provost

The work reported herein was accomplished with the support of the U.S. Army Training & Doctrine Command, Fort Monroe, VA.

Prepared by:

James K Hartman

James K. Hartman
Department of Operations Research

Reviewed by:

Michael G. Sovereign

Michael G. Sovereign, Chairman
Department of Operations Research

Released by:

William M. Tolles

William M. Tolles
Dean of Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 14 NPS55-80-021	2. GOVT ACCESSION NO. AD-A086519
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): 6 Ground Movement Modelling in the STAR Combat Model	5. TYPE OF REPORT & PERIOD COVERED: 9 Technical Report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): 10 James K. Hartman	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 15 MIPR -32CAAM01
11. CONTROLLING OFFICE NAME AND ADDRESS: U.S. Army TRADOC, Fort Monroe, VA 23651	12. REPORT DATE: 11 May 1980
	13. NUMBER OF PAGES: 57
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 12 57	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Land Combat
Combat models
Movement models
STAR

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the ground movement module used in the STAR combined arms combat simulation model. The model capabilities, data requirements, and computer programs used are presented. This report is one of a series of STAR publications.

251450

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

A

I. OVERVIEW

A. Introduction

The ground movement routines documented in this report are designed to be part of the STAR (Simulation of Tactical Alternative Responses) ground-air combat model developed at the Naval Postgradute School. STAR is a high resolution computer simulation model written in the SIMCRIPT II.5 simulation language. The ground movement routines are responsible for updating the positions of vehicles, (both combat and support) which are moving on the ground during the simulation. Separate routines will be used for air movement.

The movement routines are designed to allow a variety of different modes of movement and to interface with terrain representations of varying degrees of detail. Thus if movement is an important facet of a study, the terrain and vehicle mobility data can be incorporated at a high level of detail, while if mobility is felt to be of subsidiary importance, a less detailed representation can be used.

The movement routines in the report were first developed for use in the updated STAR Battalion model (ca. spring 1979), and have been incorporated into the expanded STAR Brigade model which is currently under development.

The remainder of the first section of the report will review the basic assumptions of the movement model and will display the various modes in which it can be used. Section II will detail the data arrays which the movement model requires. Section III will discuss the ground movement subroutines in detail, and Section IV will consider the interfacing of the movement model with the rest of STAR. Finally, Section V will briefly indicate some aspects of movement which are not developed in this report. Discussion of other parts of the STAR model can be found in references [1] thru [4].

B. Basic Assumptions of the Movement Model

The ground movement model of STAR moves vehicles between positions along routes which are made up of straight line segments. At any time in the simulation the STAR model can request a location update for any vehicle. Movement in the model is by individual vehicle. Movement control--the decision of whether to move and, if so, where--is, however, usually by some higher level unit. Thus the relationship of routes to positions is organized by platoons.

The origins and destinations within the model are represented by Position areas. A position area is a collection of individual element X, Y coordinates (e.g. a company team defensive position, or the starting position for an attacking battalion). The simulation model will support an arbitrary number of position areas.

Within each position area, the individual element positions are organized by platoon. Movement from one position area to another is along routes which are assigned for each platoon. Thus a fundamental assumption embedded in the data structure for the movement model is:

> If two vehicles, both in platoon i, are to move from
> position area A to position area B, they must use
> the same route.

As we shall see shortly, this does not require that they follow exactly the same path (because of the formation offsets), but essentially they will move in the same general location. Also, since movement is by individual vehicle, we are not required to always move the entire platoon at the same time along the same route, but often this will be the case.

Movement formations may be used to ensure that a platoon moving along a route will do so in an organized fashion (e.g. to achieve a line formation in the final assault phase of an attack: formation offsets displace the platoon members left and right from the route they are following).

Movement for each vehicle is continuous in the sense that:

i) Any x, y coordinate on the battlefield is a possible location for a unit.

ii) Arbitrarily small (or large) movement increments can be requested (e.g. move a unit for 2.2 seconds).

iii) The movement speed for a unit is continuous--if a unit wants to change speed, it can do so only gradually, limited by bounds on possible acceleration and deceleration.

iv) Acceleration however, is not continuous. Typically if a unit wishes to accelerate it will do so at the maximum allowed acceleration rate until the new speed is reached, and then will move at that new (constant) speed until it is time to change speed again.

Limitations on speed and acceleration may be derived for each vehicle from actual terrain data (for high resolution movement) or may be set to reasonable values for each vehicle throughout the simulation (for lower resolution movement).

C. Capabilities of the Model

This section summarizes the various modes in which the movement model may be used. These modes relate to the choice of origin, destination, route, and formation for the movement.

1. MOVEMENT IN PREDETERMINED DIRECTION

For simplistic analyses, the model can be used to analyze an attack where each element is given a starting position and a direction of movement. No route need be specified as movement is always along the specified direction.

2. MOVEMENT TO SPECIFIED POSITION IN A POSITION AREA

Given the current location of a vehicle, the move model can move it directly along a straight line (no route) to a specified position within a given position area. This mode would, for example, allow direct movement from one position area to another.

3. MOVEMENT TO BEST POSITION IN A POSITION AREA

Mode 3 is similar to mode 2, except that the model scans all possible positions in the vehicle's platoon area (in the given destination position area) and selects the best position which is not currently occupied.

4. MOVEMENT TO DESIGNATED ROUTE

Given any current vehicle position and a choice of route, the move model will move the vehicle from its position onto the route.

5. MOVEMENT ALONG DESIGNATED ROUTE

Given a position on a route, the model can continue along the route--(this is probably the most frequently used mode).

6. ROUTE SELECT

Given origin and destination position area numbers the model will select the route to be used and then proceed as in modes 4 and 5.

7. MOVEMENT WITH FORMATION OFFSET

In modes 5 and 6, the movement path along the route may be offset from the route to put different platoon members in different relative positions within a movement formation. The current implementation has platoon formations only.

8. FORMATION SPECIFIED BY TERRAIN

Normally a movement formation will be tactically determined. Sometimes, however, the terrain forces a particular formation (e.g. column for crossing a bridge). The move model can store and automatically implement formation changes which are required by the terrain.

9. STOP ALONG ROUTE

Upon command, the move model will stop a vehicle which is moving. This is useful, for example, if an attacking force decides to enter a hasty defense because of attrition levels.

10. REVERSE

Upon command, the move model will reverse the previous direction of movement of a vehicle and cause it to return to the position from which it began. This feature could be used if a thwarted attacking force has to retreat.

These modes are automatically combined in certain circumstances. For example, a command to move from area A to area B might automatically invoke modes

6	(to select the route)
4	(to get onto that route from area A)
5 and 7 (maybe 8)	(to move along the route in formation)
3	(to get into the best available position upon arrival in area B)

The control commands which determine the modes used will be discussed later in the report.

II. DATA REQUIREMENTS

In this section the data required to support the ground movement model is specified. As indicated in the previous section there are several different modes of movement which can be used. Depending on the mode used, the data requirements may vary. The general model which allows movement between arbitrary position areas along routes and using formation offsets requires that all of the following be available.

A. POSITION

POSITION is a 3-dimensional ragged array which gives the X and Y coordinates of potential vehicle locations (e.g. prepared defensive positions, starting attack positions). Also included is an orientation angle, Θ , for each position to allow vehicles in the defense to select their sector of responsibility. Θ is measured in radians counterclockwise from East. Positions are organized by platoons, and, for each platoon, by the position areas which that platoon might occupy.

The array POSITION (I,J,K) has subscripts

I = platoon number (I = 1,...,PNUM)

J = counter for areas; J = 1,..., number of areas this platoon might occupy

(J may be different for different platoons, I.)

For the Jth position area for Platoon I, the X, Y, Θ values are organized as follows. Typically all areas for a given platoon will have as many vehicle positions as there are elements in the platoon.

K = 1	Position Area ID number	
2	X_1	x, y, Θ for first vehicle in this platoon in this position area
3	Y_1	
4	Θ_1	
5	X_2	second position for this platoon in the position area
6	Y_2	
7	Θ_2	
	$\vdots$	
3n-1	X_n	last position for this platoon in the position area
3n	Y_n	
3n+1	Θ_n	

If the movement model is to be used in mode 3--select best position in platoon area--the individual vehicle positions 1, ... , n are assumed to be arranged in priority order. ($X_1Y_1\Theta_1$ is the best position, $X_2Y_2\Theta_2$ is next best, etc.) This mode can be important in moving to subsequent defensive positions in the active defense. If only part of a platoon survives to reach the new position area, we want them to occupy the vehicle positions which have the best defensive characteristics.

The POSITION array is initialized by the following segment of Simscript code in the RES.MOVE program given in Figure 1.

To illustrate the POSITION array and other arrays to be defined in this section, consider the following condensed and simplified movement scenario.

Platoon 1 has 3 vehicles and will fight in place in position area 37--no movement required.

Platoon 2 has 2 vehicles. It begins in position area 37 and may move either to position area 16 or to position area 42 along designated routes. (See Figure 2.)

```
ROUTINE RES.MOVE
DEFINE I, J, K, N, ID, NE, NA, NM AS INTEGER VARIABLES
USE UNIT 5 FOR INPUT
USE UNIT 6 FOR OUTPUT
LET MAX.DIST.INCR = 50.
LET FOR.CHG.INT = 200.
RESERVE POSITION(*,*,*) AS PNUM BY *  ''PNUM IS NUMBER OF PLATOONS
FOR I = 1 TO PNUM
DO
    READ ID, NE, NA  ''PLT NO., NO. ELEMS IN PLT, NO. AREAS USED BY PLT
    IF ID NOT EQUAL TO I PRINT 1 LINE WITH I AS FOLLOWS
XXXXX INPUT DATA SEQUENCE ERROR IN POSITION ARRAY FOR PLT ***** XXXXX
    ALWAYS
    RESERVE POSITION(I,*,*) AS NA BY 3*NE + 1
    FOR J = 1 TO NA
    DO
         FOR K = 1 TO 3*NE + 1
         READ POSITION(I,J,K)  ''ALL POSITION DATA FOR THIS PLATOON
         FOR K = 4 TO 3*NE+1 BY 3
         LET POSITION(I,J,K) = POSITION(I,J,K)/RADIAN.C  ''DEG TO RADIANS
    LOOP
LOOP  ''END OF INITIALIZATION FOR POSITION ARRAY
```

Figure 1. Initialization of POSITION Array

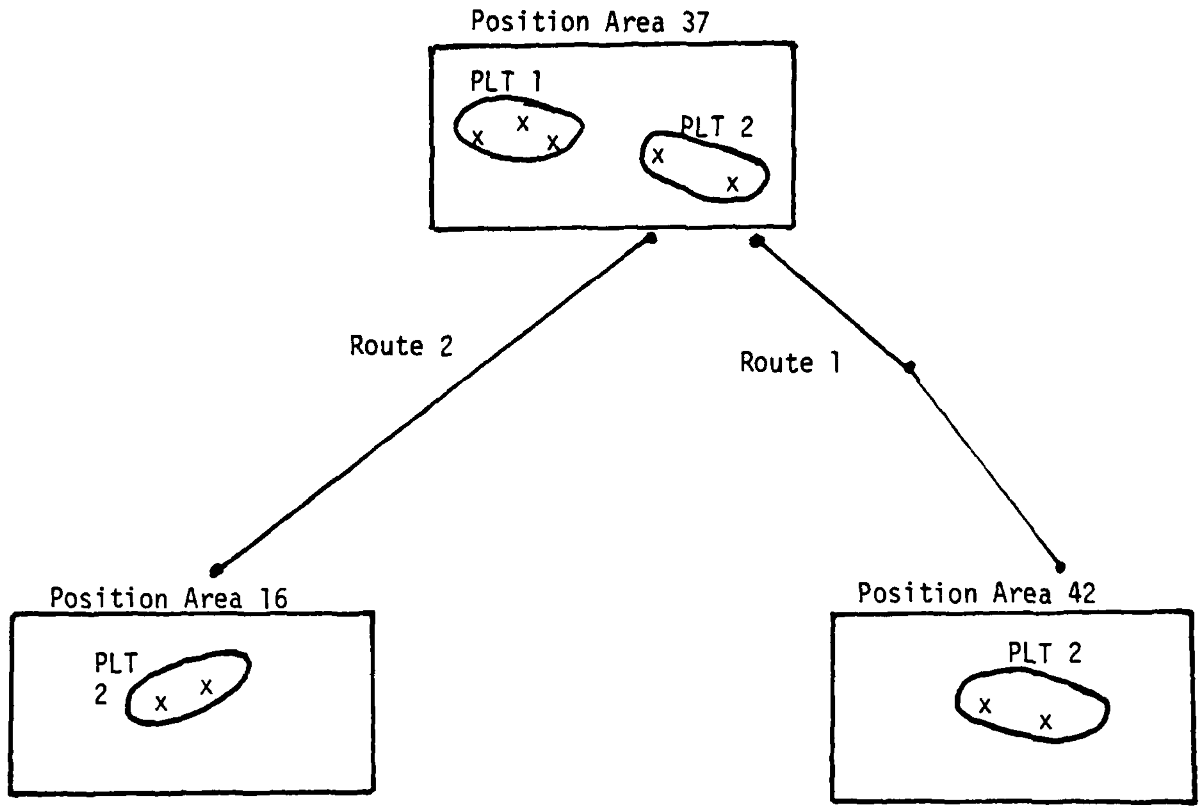


Figure 2. Simplified Movement Scenario

Input data for the POSITION array consistent with this movement scenario follows:

1 3 1	(platoon #1 has 3 vehicles and may occupy 1 position area)
37 X_1 Y_1 θ_1 X_2 Y_2 θ_2 X_3 Y_3 θ_3	(the single area is area 37 and within area 37 the 3 vehicles have X, Y, and θ coordinates)
2 2 3	(platoon #2 has 2 vehicles and may occupy 3 position areas)
37 X_4 Y_4 θ_4 X_5 Y_5 θ_5 16 X_6 Y_6 θ_6 X_7 Y_7 θ_7 42 X_8 Y_8 θ_8 X_9 Y_9 θ_9	The 3 areas are areas 37, 16 and 42. Each has 2 vehicle positions.

B. <u>MOVE.DATA</u>

The MOVE.DATA array is used to select a route between two position areas for a given platoon. The data is organized by platoon. For each platoon, MOVE.DATA contains a list of triples

$$A_1, \; A_2, \; R$$

where A_1, A_2 are area numbers and R is the number of the route connecting A_1 to A_2. For each triple it is assumed that A_1 is less than A_2, and that route R starts at A_1 and ends at A_2. If a vehicle wants to move from area A_2 to area A_1, then it will traverse route R backwards.

The MOVE.DATA array has subscripts

I = platoon number (I = 1,..., PNUM)

J = 1,..., 3 * number of area pairs for this platoon.

For platoon I, the data is organized as follows

J = 1	A_1	Area Numbers
2	A_2	
3	$R_T(1 \rightarrow 2)$	Route from A1 to A2 for platoon I
4	A_3	Area Numbers
5	A_4	
6	$R_T(3 \rightarrow 4)$	Route from A3 to A4 for platoon I
⋮		

The (A_1, A_2) pairs in the list are assumed ordered lexicographically in increasing order to aid the search process in route selection (e.g. if the area pairs to be connected are

$$(1,2), \quad (1,4), \quad (2,4), \quad (3,2)$$

then their order in the list is as written above).

Note that different platoons may use different routes between the same area pairs. Also, that only those area pairs which are used by a platoon will be included in its section of the MOVE.DATA Array.

MOVE.DATA is initialized in the RES.MOVE program by the code segment in Figure 3.

A typical set of input data, consistent with the previous POSITION data follows: (again PNUM = 2 = # of platoons)

```
 1   0                            platoon 1 has only one area, so it
                                  will never move--zero routes used.
 2   2                            platoon 2 has 2 movement possibilities
16   37   2   37   42   1
```

Platoon 2 will use route 2 to get from area 16 to area 37 or route 1 to get from 37 to 42.

```
RESERVE MOVE.DATA(*,*) AS PNUM BY *  ''PNUM IS NUMBER OF PLATOONS
FOR I = 1 TO PNUM
DO
    READ ID, NM    ''PLATOON NO., NO. OF ROUTES USED BY THIS PLT
    IF ID NOT EQUAL TO I PRINT 1 LINE WITH I AS FOLLOWS
XXXXX INPUT DATA SEQUENCE ERROR IN MOVE.DATA ARRAY FOR PLT ***** XXXXX
    ALWAYS
    IF NM EQUALS 0   CYCLE
    ELSE
    RESERVE MOVE.DATA(I,*) AS 3*NM
    FOR J = 1 TO 3*NM
    READ MOVE.DATA(I,J)
LOOP   ''END OF INITIALIZATION FOR MOVE.DATA ARRAY
```

Figure 3. Initialization of MOVE.DATA Array

There could also be a route from 16 to 42 but in this example we have (arbitrarily) decided that the simulation scenario does not require it. If it were included, the lexicographic ordering would require that it be between the above two triples. Note also that the route runs from 16 to 37 even though the platoon will be moving from 37 to 16.

C. ROUTE.DATA

The ROUTE.DATA array contains the description of each of the routes used by the model. A route is a sequence of X, Y coordinates called movement control points (MCPS). Between MCPS the model plots straight line route segments.

Subscripts for the array ROUTE.DATA(I, J) are

I = route number, I = 1,...,number of routes

for each route I, the J subscript indexes a list of coordinate pairs of MCP's.

Along with each MCP coordinate pair is stored a platoon formation code. If this code is 0 it means any formation can be used on this route segment. Otherwise the indicated formation must be used.

J = 1	X_1	First MCP
2	Y_1	
3	form code_1	Refers to segment between MCP's 1 and 2
4	X_2	Second MCP
5	Y_2	
6	form code_2	Refers to segment between MCP's 2 and 3
⋮		

The number of MCP's can be different for different routes, and many different platoons may use the same route. ROUTE.DATA is initialized in the RES.MOVE program by the code segment in Figure 4.

```
READ N              ''NUMBER OF ROUTES TO USE
RESERVE ROUTE.DATA(*,*) AS N         BY *
FOR I = 1 TO N
DO
    READ ID, NM     ''ROUTE NUMBER, NO. OF MVMT CONTROL POINTS ON THIS ROUTE
    IF ID NOT EQUAL TO I PRINT 1 LINE WITH I AS FOLLOWS
XXXXX INPUT DATA SEQUENCE ERROR IN ROUTE.DATA ARRAY FOR ROUTE ***** XXXXX
    ALWAYS
    RESERVE ROUTE.DATA(I,*) AS 3*NM
    FOR J = 1 TO 3*NM
    READ ROUTE.DATA(I,J)
LOOP      ''END OF INITIALIZATION FOR ROUTE.DATA ARRAY
```

Figure 4. Initialization of ROUTE.DATA Array

A typical data input is as follows (again consistent with our previous scenario).

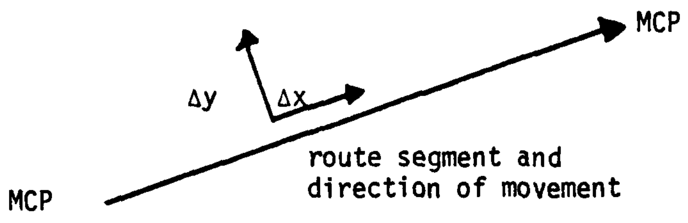
2	number of routes in the array
1 3	route #1 has 3 MCP's
X_1 Y_1 0 X_2 Y_2 1 X_3 Y_3 0	(3 MCP coordinates, formation 1 must be used between MCP2 and MCP 3)
2 2	route #2 has 2 MCP's
X_4 Y_4 0 X_5 Y_5 0	(2 MCP coordinates, no formation requirement)

(Note that the X,Y coordinates here are different numbers from those in the POSITION array, even though we have used the same notation for both).

D. FORM.OFFSET

The FORM.OFFSET array contains information on the relative position of vehicles in a formation expressed as offsets from the route.

The offsets are assumed to be



Δx positive in direction of movement, and

Δy positive left of the route.

The FORM.OFFSET(I,J) array has subscripts,

I = 1,..., number of formations

J = 1,..., 2 * number of places in the formation (may be different for different I)

and for formation I, the offsets are organized as

J = 1	Δx_1	offsets for first place
2	Δy_1	in formation
3	Δx_2	second place
4	Δy_2	
	⋮	etc.

FORM.OFFSET is initialized by the final segment of code in RES.MOVE given in Figure 5.

A typical input data set is:

2	number of formations
1 2 0 0 -50 0	offsets for 2 places in formation 1; place 2 is 50 meters behind place 1; both are on the route in column.
2 4	
0 -150 0 -50 0 50 0 150	(a 4 place line formation with 100 meter spacing)

E. Vehicle Attributes

Each vehicle in the STAR simulation is a temporary entity which has numerous attributes attached to it. The attributes which are of interest to the move routine are listed below.

MV.STATE — the primary control variable for initiating and stopping movement. possible values and their meanings are:
- 0 - in position. Do not move.
- 1 - want to start movement from one position area to another--do a route selection and start to move
- 2 - continue movement along a previously selected route
- 3 - stop along route (e.g. stop to fire)

```
READ N               ''NUMBER OF MOVEMENT FORMATIONS
RESERVE FORM.OFFSET(*,*) AS N        BY *
FOR I = 1 TO N
DO
    READ ID,NM      ''FORMATION NO., NO. OF PLACES IN THAT FORMATION
    IF ID NOT EQUAL TO I PRINT 1 LINE WITH I AS FOLLOWS
XXXXX INPUT DATA SEQUENCE ERROR IN FORM.OFFSET ARRAY FOR FORMATION ***** XXXXX
    ALWAYS
    RESERVE FORM.OFFSET(I,*) AS 2*NM
    FOR J = 1 TO 2*NM
    READ FORM.OFFSET(I,J)
LOOP      ''END OF INITIALIZATION FOR FORM.OFFSET ARRAY
RETURN
END
```

Figure 5. Initialization of FORM.OFFSET Array

	4 - next position has been reached--stop.
	5 - final position has been reached--never move again.
AREA.START	The origin and destination position area
AREA.END	numbers for the movement of a vehicle.
ROUTE	The route number along which the vehicle moves. 0 if not using routes.
NEXT.MCP	MCP number on designated ROUTE toward which the vehicle is now moving. 0 if end of route has been reached.
DIR.ON.RT	0 if vehicle is moving in direction of increasing MCP numbers along route 1 if vehicle is traversing MCP's in decreasing order (backwards along route)
X.CURRENT	location of vehicle
Y.CURRENT	at most recent
Z.CURRENT	movement update
SPD	speed of vehicle at end of most recent movement update
T.SPD	simulation time at which most recent movement update ended. (Time SPD was last set.)
DIR.OF.MVMT	angular direction of movement (measured in radians from East)
PLT	the platoon to which the vehicle belongs
POS.IN.PLT.AREA	position number indicating which position in the POSITION array this vehicle is occupying or plans to occupy. (Zero while on route if best position is to be chosen on arrival in position area.)
FORM.CODE	formation number for the platoon (0 if not in formation; then vehicle moves along route without offset).
FORM.POS	number indicating which place in the formation this vehicle should occupy.

F. Set Membership

The vehicle temporary entities may belong to several sets. For movement control purposes the most important set is the PLT.UNIT set which is owned by a PLATOON.LEADER. Since position areas, routes, and formations are organized by platoon, it is frequently necessary to reference a vehicle's platoon and the other elements in that platoon by looping through the PLT.UNIT set.

III. MODEL SUBROUTINES

In this section we review the subroutines which are used for executing ground movement commands in the STAR model. There are several quite brief "utility" routines, and one quite lengthy MOVE routine. For each routine we list the local variables, global variables, routines called, events scheduled, a brief description of the code, and information on how to use the routine.

A. Routine INIT.POS

The INIT.POS routine selects the initial position of a given vehicle from the POSITION array.

Input Argument

VEH	pointer of the vehicle to be positioned

Local Variables

I, J, K	array subscripts

Global Variables Used

POSITION	array of positions

Vehicle Attributes Used

PLT	platoon number
POS.IN.PLT.AREA	position number for the vehicle
AREA.START	the area in which to place this vehicle
X.CURRENT Y.CURRENT	resulting X and Y coordinates from POSITION
Z.CURRENT	elevation coord from ELEV
DIR.OF.MVMT PRI.DIR	} both set to resulting θ orientation angle from POSITION

Routines called

BEST.POS(VEH)
ELEV

Events scheduled

none

Code - See Figure 6.

Brief Description

Line 5	IF POS.IN.PLT.AREA is zero, then routine BEST.POS is called to select the best available position (this call sets POS.IN.PLT.AREA to a nonzero value).
Line 8	scans the list of position areas for this platoon until a match is found with AREA.START
Lines 9-13	then select the desired position from this area and set the vehicle's X and Y coordinates to this position. The vehicle's movement and search orientations are also set.
Line 14	calls the ELEV routine to set the vehicle's Z coordinate from the terrain.

Use

Enter with

i) vehicle pointer

ii) PLT

iii) AREA.START

iv) (optional) POS.IN.PLT.AREA

On Exit, routine has set

i) POS.IN.PLT.AREA (if zero on entry)

ii) XCURRENT, Y.CURRENT,Z.CURRENT

iii) DIR.OF.MVMT, PRI.DIR

```
ROUTINE FOR INIT.POS(VEH)
'' SELECTS INITIAL POSITION FOR AN ELEMENT
DEFINE VEH, I, J, K AS INTEGER VARIABLES
LET I = PLT(VEH)
IF POS.IN.PLT.AREA(VEH) EQUALS 0 CALL BEST.POS(VEH)
ALWAYS
LET K = POS.IN.PLT.AREA(VEH) * 3
FOR J = 1 TO DIM.F(POSITION(I,*,*)) WITH POSITION(I,J,1) EQUALS AREA.START(VEH)
DO
    LET X.CURRENT(VEH) = POSITION(I,J,K-1)
    LET Y.CURRENT(VEH) = POSITION(I,J,K)
    LET DIR.OF.MVMT(VEH) = POSITION(I,J,K+1)
    LET PRI.DIR(VEH) = DIR.OF.MVMT(VEH)
CALL ELEV(X.CURRENT(VEH),Y.CURRENT(VEH)) YIELDING Z.CURRENT(VEH)
LOOP
RETURN
END
```

Figure 6. Routine INIT.POS

B. Routine BEST.POS

The BEST.POS routine selects the best (first) available position number for a vehicle to occupy in the platoon's position area.

Input Argument

VEH	pointer of the vehicle to be positioned

Local Variables

ELEM	pointer to other vehicles in VEH's platoon
J	the position number

Global Variables Used

none

Vehicle Attributes Used

PLT	platoon number
POS.IN.PLT.AREA	position number to be selected by the routine, also position number for other vehicles

Routines Called

None

Events Scheduled

None

Sets Used

PLT.UNIT	the platoon to which the vehicle belongs.

Code - see Figure 7.

Brief Description

Line 6	considers J = 1,2,..., number of vehicles in this platoon
Lines 8-13	for each such J, loop over all elements in the platoon. If any element has a POS.IN.PLT.AREA = J, then position J is already occupied and thus not available for this vehicle. Hence go to 'NEXT.POS' to try the next J value.
Lines 14-15	If position J is available--use it and return.

```
ROUTINE FOR BEST.POS GIVEN VEH
'' CALLED WHEN VEHICLE REACHES END OF MOVEMENT ROUTE CLOSE TO NEW
'' POSITION AREA.  CHOOSES BEST EMPTY POSITION IN HIS PLATOON AREA
'' FOR HIM TO OCCUPY.
DEFINE VEH, J, ELEM AS INTEGER VARIABLES
FOR J = 1 TO N.PLT.UNIT(PLT(VEH))   ''NO. ELEMENTS IN PLT SET TO WHICH VEH BELONG
DO
    FOR EACH ELEM IN PLT.UNIT(PLT(VEH))
    DO
         IF POS.IN.PLT.AREA(ELEM) EQUALS J ''POSITION J IS ALREADY FULL
              GO TO NEXT.POS
         ELSE
    LOOP  '' TO SEE IF NEXT ELEMENT OF PLT IS IN POS J
LET POS.IN.PLT.AREA(VEH) = J
RETURN
'NEXT.POS'
LOOP   ''BACK TO TRY NEXT BEST POSITION
END
```

Figure 7. Routine BEST.POS

Use

Enter with vehicle pointer.

On entry, POS.IN.PLT.AREA(VEH) should be zero.

On exit POS.IN.PLT.AREA will have been set to an integer J which is the first available position number. If the positions are stored in prioritized order, the first available position will be the best available.

C. Routine RT.SEL

The RT.SEL routine selects the route to be used for a given movement and sets the ROUTE, NEXT.MCP, and DIR.ON.RT attributes of the vehicle to correspond to this route:

Input Variables

VEH	pointer to vehicle

Local Variables

A1	position area numbers defining
A2	the movement desired
P	platoon number
I, J	array subscript
AREA	position area number

Global Variables Used

MOVE.DATA

Vehicle Attributes Used

PLT	
AREA.START	input
AREA.END	
ROUTE	
DIR.ON.RT	output
NEXT.MCP	

Routines Called

None

Events Scheduled

None

Code - see Figure 8.

Brief Description

Lines 5-6	define (A1,A2) as the area number pair with A1 < A2
Lines 9-15	search the MOVE.DATA array to find the pair (A1,A2)
Line 16	sets ROUTE
Lines 17-20	set NEXT.MCP and DIR.ON.RT to indicate forward movement along Route
Lines 21-25	set NEXT.MCP and DIR.ON.RT to indicate backward movement along Route
Lines 8 and 27	if no match is found to (A1,A2) the ROUTE =0 on return.

Use

RT.SEL should only be used once at the start of a movement between areas (MV.STATE = 1) because it sets NEXT.MCP to the closest end of the route regardless of the actual position of the vehicle.

It is called automatically in the MOVE routine whenever MV.STATE = 1.
On entry, the AREA.START and AREA.END attributes completely define the desired move.
On exit, the ROUTE, NEXT.MCP, DIR.ON.RT attributes have been set to define details of the desired move.

D. ROUTINE MOVE.LIMITS

The MOVE.LIMITS subroutine is responsible for determining limits on the speed and acceleration with which the vehicle can move. Several different versions of this routine can be written, depending on the degree of resolution desired for movement.

```
ROUTINE FOR RT.SEL GIVEN VEH
'' CALLED WHEN VEHICLE FIRST LEAVES A POSITION AREA.  CHOOSES THE ROUTE
'' ALONG WHICH VEH WILL MOVE TO REACH NEXT POSITION AREA
DEFINE VEH,A1,A2,P,I,J,AREA AS INTEGER VARIABLES
LET A1 = MIN.F(AREA.START(VEH),AREA.END(VEH))
LET A2 = MAX.F(AREA.START(VEH),AREA.END(VEH))
LET P = PLT(VEH)
LET ROUTE(VEH) = 0
FOR I = 1 TO DIM.F(MOVE.DATA(P,*))/3
DO
    LET J = 3*I
    LET AREA = MOVE.DATA(P,J-2)
    IF AREA IS GREATER THAN A1 RETURN  ''NO MATCH FOR AREA NUMBERS FOUND
    ELSE
    IF AREA EQUALS A1 AND MOVE.DATA(P,J-1) EQUALS A2
         LET ROUTE(VEH) = MOVE.DATA(P,J)
         IF AREA.START(VEH) IS LESS THAN AREA.END(VEH)  ''NORMAL DIRECTION
              LET DIR.ON.RT(VEH) = 0
              LET NEXT.MCP(VEH) = 1
              RETURN     ''WITH FORWARD ROUTE
         ELSE
              LET DIR.ON.RT(VEH) = 1
              LET NEXT.MCP(VEH) = DIM.F(ROUTE.DATA(ROUTE(VEH),*))/3
              RETURN     ''WITH BACKWARD ROUTE
    ELSE
LOOP     ''BACK TO TRY NEXT SEGMENT OF MOVE.DATA ARRAY
RETURN   ''WITH NO ROUTE -- MATCH NOT FOUND
END
```

Figure 8. Routine RT.SEL

A high resolution movement model might consider digitized limiting speed and acceleration values based on extensive terrain analysis and on detailed vehicle characteristics.

A low resolution movement model might set these values constant for each vehicle type independent of terrain details; other resolutions are not difficult to imagine.

In any case, the influence of terrain on movement can be concentrated in the MOVE.LIMITS routine thus enhancing the flexibility of the model. In this report we will not detail the MOVE.LIMITS routine as it depends so much on the terrain representation chosen. The general characterisitcs of the routine are, however,

Input Variables

VEH	Pointer to the vehicle
SLOPE	*Dimensionless terrain slope(rise ÷ run)*

Output Variables

SPEED	The target speed which the movement model should try to match in this move increment. This speed may depend on terrain, the unit's desired maneuver speed, obstacles or minefields, etc. In particular, if the vehicle's MV.STATE is 3, SPEED should be set to 0 (Stop).
ACCEL	(> 0) limit on allowed acceleration of the vehicle. Again this may be modelled in varying degrees of detail.
DECEL	(< 0) limit on allowed deceleration.

Calling Sequence

CALL MOVE.LIMITS(VEH, SLOPE) YIELDING SPEED, ACCEL, DECEL

Called From

MOVE	at start of each movement increment

E. Routine ELEVG

Routine ELEVG provides the macro terrain representation for the simulation. For any X, Y coordinates on the battlefield it provides the Z (vertical elevation) coordinate and the gradient components GX and GY of the terrain. ELEVG is similar to MOVE.LIMITS in that it is independent of the rest of the move model and can be realized in several different ways depending on the study requirements. For example, tabletop terrain is obtained by having ELEVG return a constant elevation and zero gradient. The current STAR implementation utilizes functionally coded terrain, and an ELEVG routine has been written for that representation. Changing to a digitized terrain representation merely requires that the appropriate routine be named ELEVG and dropped into place. It is important to note that this (and any other) routine may be written in either FORTRAN or SIMSCRIPT.

<u>Input Variables</u>

X Y	map coordinates

<u>Output Variables</u>

Z	elevation
GX GY	gradient components

<u>Attributes Used</u>

None

<u>Routines Called</u>

None

<u>Events Scheduled</u>

None

<u>Code</u> See other STAR publications[2] for the currently used functional terrain model and the corresponding ELEVG program.

F. Routine MOVE

The MOVE routine updates the location, direction, and speed of a vehicle to the current simulation time. The routine is rather long, but not overwhelmingly complex. It is useful to consider it as five sequential parts which are almost always executed in strict sequence:

i) compute a destination point

ii) compute direction and angles from current location to destination point

iii) relate distance, time, speed, and acceleration to define the move increment

iv) update location and time for this move increment

v) check whether finished. If not, go back to i) or iii).

Parts i) and iii) comprise the bulk of the code because each contains several alternative possible computational sequences. Part i) determines the various modes of movement as listed earlier in this report, and Part iii) must consider move increments limited by either time or distance for any given speed and acceleration limits. In describing the logic of the move routine we will use this breakdown by the 5 listed program segments.

Input Variable

VEH

Local Variables

numerous	all are listed in define statements at beginning of program--we explain them as appropriate in discussing program logic.

Global Variables and Arrays

POSITION	positions in areas
ROUTE.DATA	MCP coordinates

FORM.OFFSET	formations
FOR.CHG.INT	distance within which formation changes are accomplished
MAX.DIST.INCR	max distance to move before re-evaluating terrain.

Routines Called

RT.SEL

BEST.POS

ELEVG

MOVE.LIMITS

Events Scheduled

None

Code

We break the code into several segments and discuss each in turn.

Segment 0) Declarations, Initialization, and MV.STATE filters - see Fig. 9.

Lines 19-22 if MV.STATE is 1 we do a route select (and set MV.STATE = 2 for all future calls to move along this route)

Lines 23 the time interval over which this vehicle's movement is to be computed runs from T.SPD to TIME.V (current simulation time).

Segment i) Compute a Destination Point

The various modes of movement are computed in this segment of the code. The result of the computation is a destination point X.DEST, Y.DEST toward which the vehicle will move.

The flowchart of Figure 10 describes the possible alternatives. The Code, given in Figure 12, is quite long because of the several distinct alternatives.

```
ROUTINE TO MOVE GIVEN VEH
DEFINE K AS AN INTEGER VARIABLE
DEFINE SLOPE AS A REAL VARIABLE
DEFINE REM.MOVE.TIME, DEL.X, DEL.Y, D.TO.MCP, ALPH,  SALPH,
    CALPH, GRAD.X, GRAD.Y, SPD.LIMIT, ACCEL.LIMIT, DECEL.LIMIT,
    DIST.LIMIT, DEL.SPD, DIST.INCR, TIME.INCR AS REAL VARIABLES
DEFINE DIST.REQ, TIME.REQ, ZERO.LEVEL AS REAL VARIABLES
DEFINE X.DEST, Y.DEST, DIR, CX, CY, NX, NY, LX, LY, NLX, NLY, PX,PY,
     NPX, NPY, X.OFF, Y.OFF, D.TO.DEST AS REAL VARIABLES
DEFINE THETA, CTH, STH AS REAL VARIABLES
DEFINE VEH, FINAL AS INTEGER VARIABLES
DEFINE MST, RT, NM, MCP.INC, LM, MCP, D.ON.RT AS INTEGER VARIABLES
DEFINE FAKE.MCP AS AN INTEGER VARIABLE
DEFINE I, J AS INTEGER VARIABLES
LET ZERO.LEVEL = 1.0     LET FINAL = 0
LET MST = MV.STATE(VEH)
IF MST EQ 0 OR MST GE 4 RETURN
ELSE
IF MST EQUALS 1
    CALL RT.SEL(VEH)
    LET MV.STATE(VEH) = 2
ALWAYS
LET REM.MOVE.TIME = TIME.V - T.SPD(VEH)
```

Figure 9. MOVE Routine, Segment 0

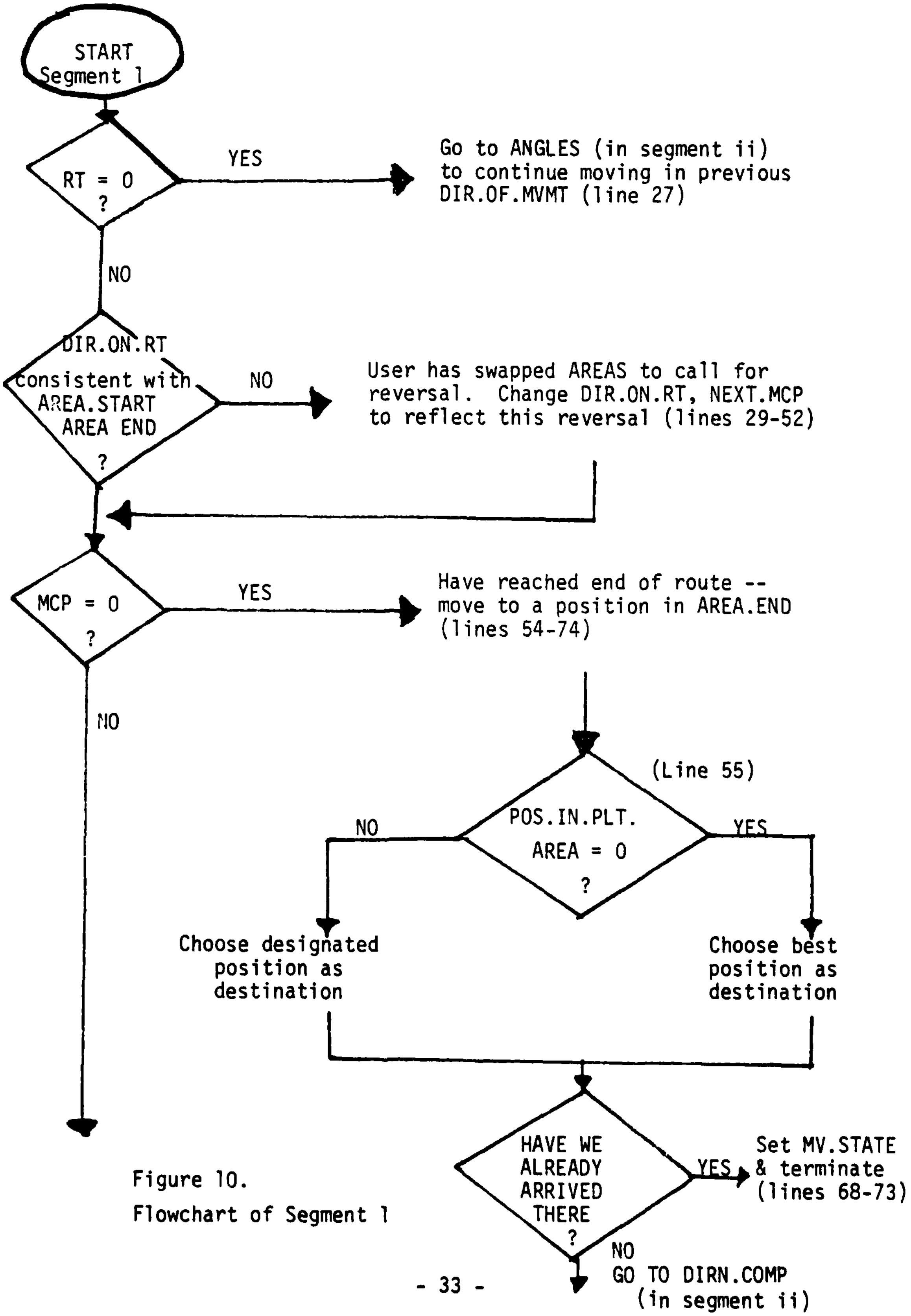


Figure 10.
Flowchart of Segment 1

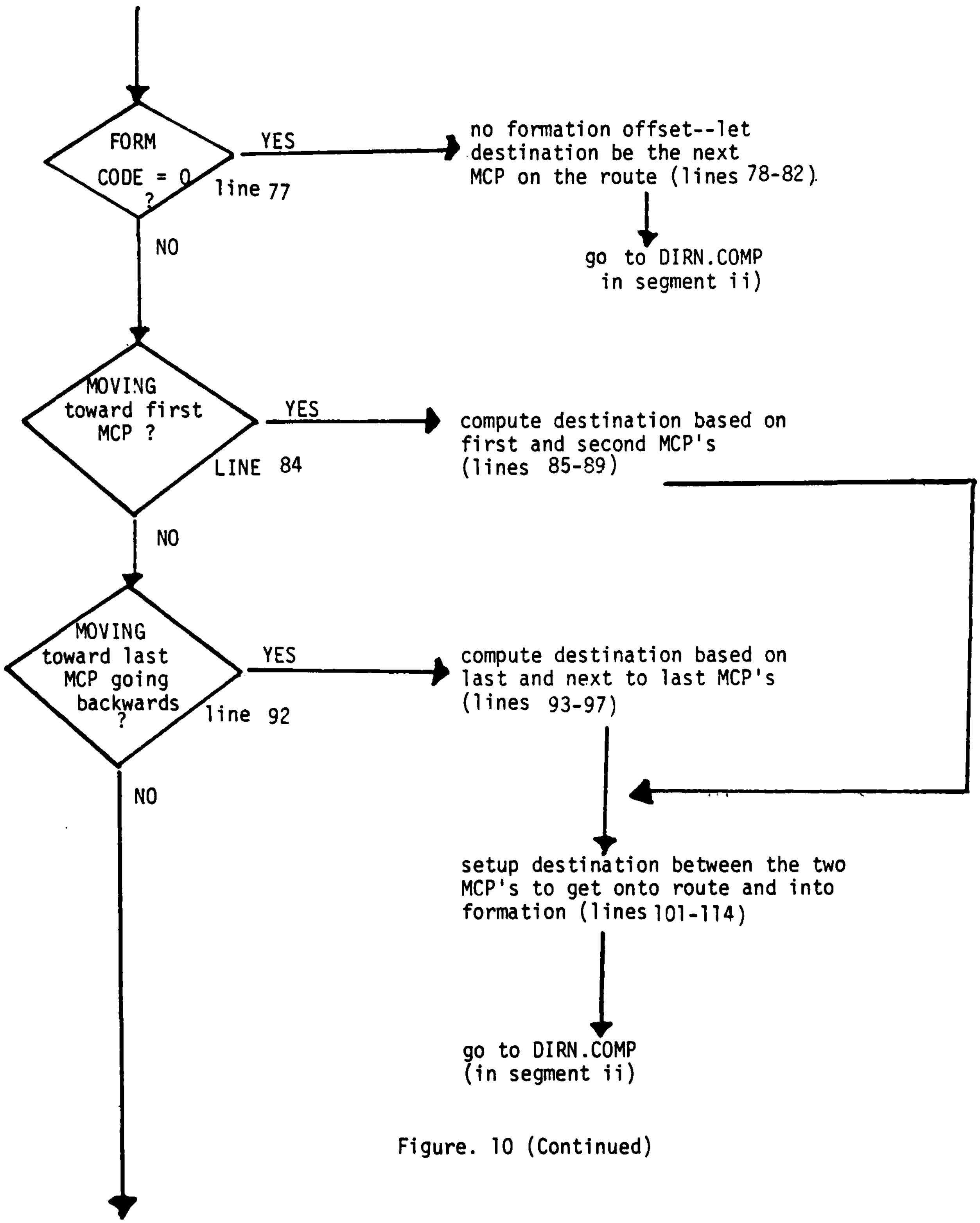


Figure. 10 (Continued)

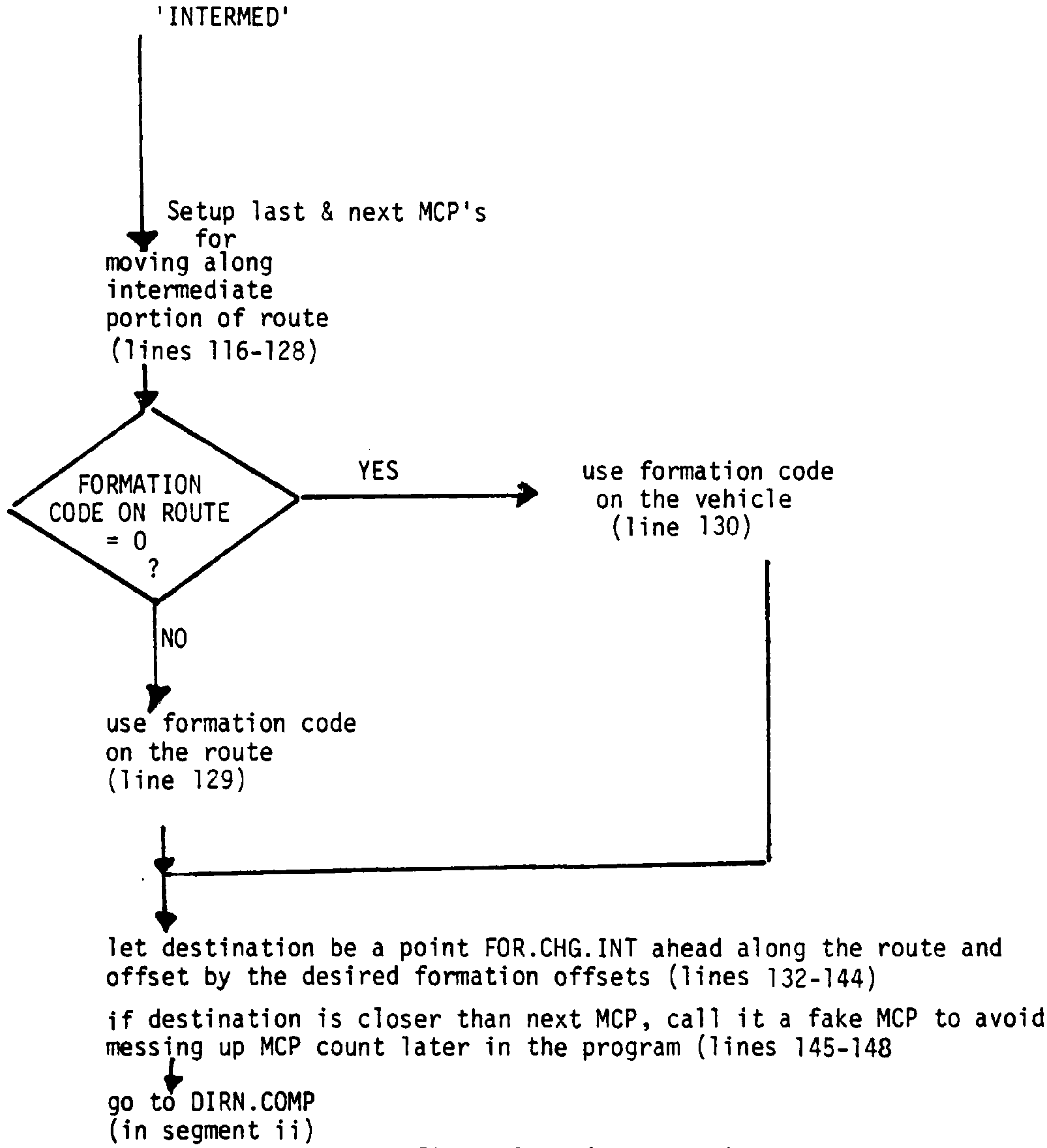


Figure 10. (Continued)

The destination point computations are a bit intricate, especially in lines 116-148 Reference to Figure 11 will clarify the several intermediate variables used.

In each case, except where RT= 0, we leave this code segment with a destination point X.DEST, Y.DEST and a distance D.TO.MCP which is the distance we can move before having to recompute a new destination point. The

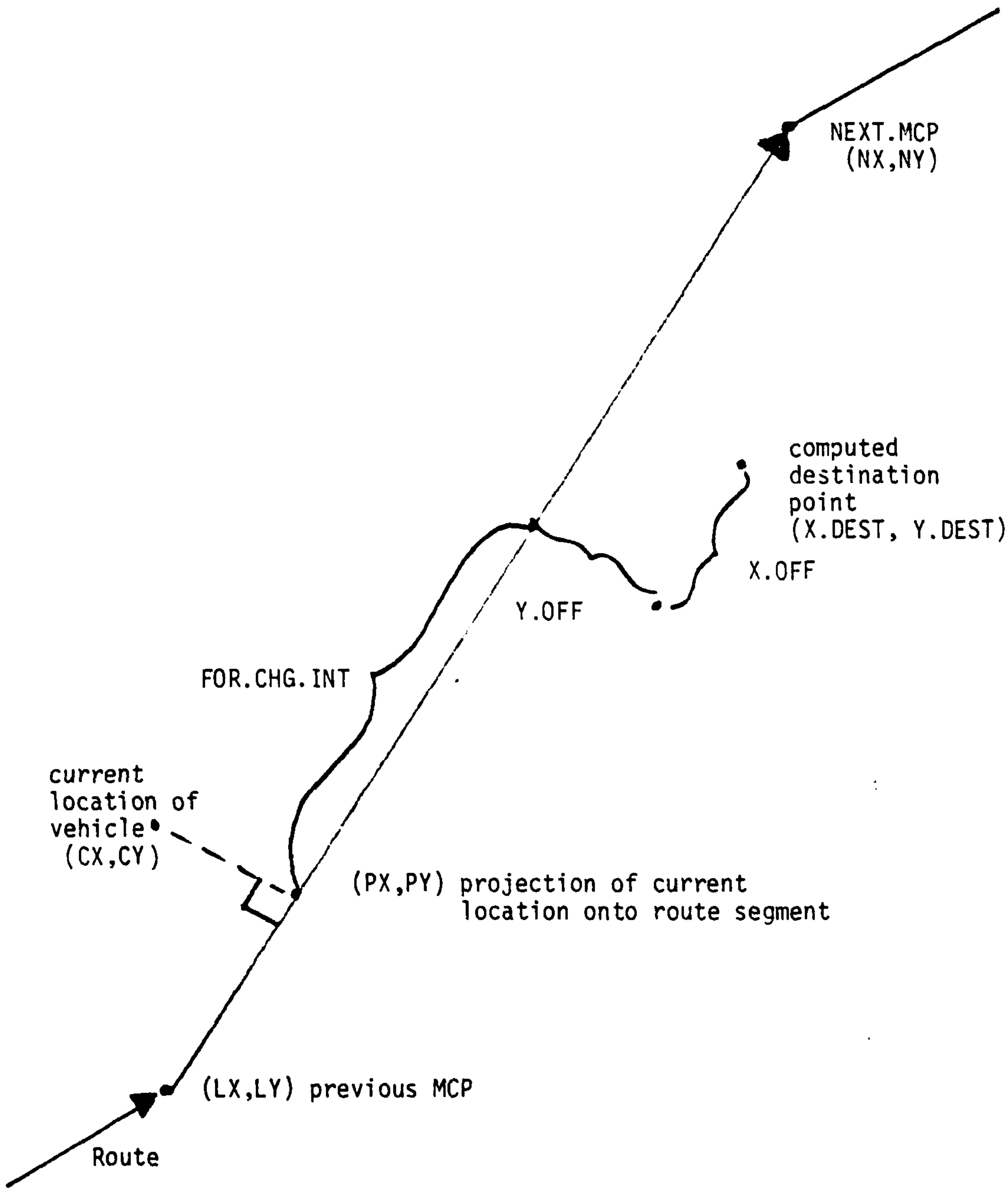


Figure 11. Geometry of Destination Computation in Intermediate Case

```
LET RT = ROUTE(VEH)
LET D.ON.RT = DIR.ON.RT(VEH)
LET FAKE.MCP = 0
IF RT EQUALS 0    LET D.TO.MCP = RINF.C    GO TO ANGLES
ELSE
''CONSISTENCY CHECK FOR POSSIBLE TURNAROUND
IF AREA.START(VEH) LT AREA.END(VEH)
    IF D.ON.RT EQ 0  GO TO NEW.MCP
    ELSE LET D.ON.RT = 0
    LET DIR.ON.RT(VEH) = 0
    LET K = DIM.F(ROUTE.DATA(RT,*))/3
    LET MCP = NEXT.MCP(VEH)
    IF MCP EQ 0 LET NEXT.MCP(VEH) = 1
    ELSE IF MCP EQ K  LET NEXT.MCP(VEH) = 0
        ELSE LET NEXT.MCP(VEH) = MCP + 1
        ALWAYS
    ALWAYS
ELSE ''AREA.START GT AREA.END
    IF D.ON.RT EQ 1 GO TO NEW.MCP
    ELSE LET D.ON.RT = 1
    LET DIR.ON.RT(VEH) = 1
    LET K = DIM.F(ROUTE.DATA(RT,*))/3
    LET MCP = NEXT.MCP(VEH)
    IF MCP EQ 0 LET NEXT.MCP(VEH) = K
    ELSE IF MCP EQ 1 LET NEXT.MCP(VEH) = 0
        ELSE LET NEXT.MCP(VEH) = MCP -1
        ALWAYS
    ALWAYS
ALWAYS
'NEW.MCP' LET MCP = NEXT.MCP(VEH)     LET NM = MCP * 3
IF MCP EQUALS 0  ''MOVE TO POSITION IN AREA.END
    IF POS.IN.PLT.AREA(VEH) EQUALS 0, CALL BEST.POS(VEH)   ''SETTING POS.IN.PLT.A
    ALWAYS
    LET I = PLT(VEH)      LET K = POS.IN.PLT.AREA(VEH) * 3
    FOR J = 1 TO DIM.F(POSITION(I,*,*)) WITH POSITION(I,J,1) EQUALS
         AREA.END(VEH)
    DO
         LET X.DEST = POSITION(I,J,K-1)
         LET Y.DEST = POSITION(I,J,K)
         LET DIR = POSITION(I,J,K+1)
    LOOP
    LET D.TO.MCP = SQRT.F((X.DEST-X.CURRENT(VEH))**2 +
         (Y.DEST-Y.CURRENT(VEH))**2)
    IF D.TO.MCP LESS THAN ZERO.LEVEL,
         LET MV.STATE(VEH) = 4
         LET DIR.OF.MVMT(VEH) = DIR
         LET PRI.DIR(VEH) = DIR
         LET SPD(VEH) = 0.
         LET FINAL = 1
         GO TO NEW.INCR
```

Figure 12. MOVE Routine, Segment i

```
        ELSE
        GO TO DIRN.COMP
    ELSE
    IF FORM.CODE(VEH) EQUALS 0       ''GO DIRECTLY TO NEXT MCP
        LET X.DEST = ROUTE.DATA(RT,NM-2)
        LET Y.DEST = ROUTE.DATA(RT,NM-1)
        LET D.TO.MCP = SQRT.F((X.DEST-X.CURRENT(VEH))**2 +
             (Y.DEST-Y.CURRENT(VEH))**2)
        GO TO DIRN.COMP
    ELSE     ''MOVE ALONG ROUTE OFFSET BY FORMATION
    IF MCP EQUALS 1 AND D.ON.RT EQUALS 0  ''TOWARD FIRST MCP
             LET NX = ROUTE.DATA(RT,4)
             LET NY = ROUTE.DATA(RT,5)
             LET LX = ROUTE.DATA(RT,1)
             LET LY = ROUTE.DATA(RT,2)
             LET I = ROUTE.DATA(RT,3)
    ELSE
        LET K = DIM.F(ROUTE.DATA(RT,*))
        IF MCP EQUALS K/3 AND D.ON.RT EQUALS 1   ''TOWARD LAST MCP GOING BACKWARD
             LET NX = ROUTE.DATA(RT,K-5)
             LET NY = ROUTE.DATA(RT,K-4)
             LET LX = ROUTE.DATA(RT,K-2)
             LET LY = ROUTE.DATA(RT,K-1)
             LET I = ROUTE.DATA(RT,K-3)
        ELSE GO TO INTERMED
        ALWAYS
    ALWAYS
        LET NLX = NX-LX       LET NLY = NY-LY
        IF I EQUALS 0,  LET I = FORM.CODE(VEH)
        ALWAYS
        LET J =    FORM.POS(VEH) * 2
        LET X.OFF = FORM.OFFSET(I,J-1)
        LET Y.OFF = FORM.OFFSET(I,J)
        LET THETA = ARCTAN.F(NLY,NLX)
        LET CTH = COS.F(THETA)
        LET STH = SIN.F(THETA)
        LET X.DEST = LX + (X.OFF + FOR.CHG.INT)*CTH  - Y.OFF*STH
        LET Y.DEST = LY + (X.OFF + FOR.CHG.INT)*STH + Y.OFF * CTH
        LET D.TO.MCP = SQRT.F((X.DEST-X.CURRENT(VEH))**2 +(Y.DEST-Y.CURRENT(VEH))
             **2)
        GO TO DIRN.COMP
    'INTERMED'      ''TO HERE FOR INTERMEDIATE MCP'S ON ROUTE
        LET CX = X.CURRENT(VEH)      LET CY = Y.CURRENT(VEH)
        IF D.ON.RT ECUALS 0  LET LM = NM - 3
        ELSE LET LM = NM + 3
        ALWAYS
        LET NX = ROUTE.DATA(RT,NM-2)
        LET NY = ROUTE.DATA(RT,NM-1)
        LET LX = ROUTE.DATA(RT,LM-2)
        LET LY = ROUTE.DATA(RT,LM-1)
```

Figure 12. (Continued)

```
LET NLX = NX - LX          LET NLY = NY - LY
LET ALPH =-((CX-NX)*NLX + (CY-NY)*NLY) / (NLX*NLX + NLY*NLY)
LET PX = ALPH * LX + (1. - ALPH) * NX
LET PY = ALPH * LY + (1. - ALPH) * NY
LET NPX = NX - PX          LET NPY = NY - PY
LET I = ROUTE.DATA(RT,NM+3*(D.ON.RT-1))
IF I EQUALS 0,  LET I = FORM.CODE(VEH)
ALWAYS
LET J =    FORM.POS(VEH) * 2
LET X.OFF = FORM.OFFSET(I,J-1)
LET Y.OFF = FORM.OFFSET(I,J)
LET D.TO.MCP = SQRT.F(NPX*NPX + NPY*NPY)
IF D.TO.MCP LESS THAN ZERO.LEVEL   GO TO MCP.REACHED
ELSE
LET THETA = ARCTAN.F(NLY,NLX)
LET CTH = COS.F(THETA)
LET STH = SIN.F(THETA)
LET ALPH = FOR.CHG.INT / D.TO.MCP
LET X.DEST = PX + ALPH * NPX + X.OFF  * CTH - Y.OFF * STH
LET Y.DEST = PY + ALPH * NPY + Y.OFF * CTH + X.OFF * STH
LET D.TO.DEST = SQRT.F((X.DEST-CX)**2 + (Y.DEST -CY)**2)
IF D.TO.DEST IS LESS THAN D.TO.MCP
     LET D.TO.MCP = D.TO.DEST
     LET FAKE.MCP = 1
ELSE LET FAKE.MCP = 0
ALWAYS
```

Figure 12. (Continued)

destination point may be an MCP, a position in a position area, or, in the case of movement in formation, a so-called fake-MCP, created temporarily to get the vehicle into the proper formation position.

Segment ii) Compute direction and angles

The brief code segment given in Figure 13 determines the DIR.OF.MVMT to the destination point and computes some trig functions for later use in segment iv).

Segment iii) Relate distance, time, speed and acceleration.

Segment iii compares the remaining move time to the distance to be moved in this increment. Depending on the desired target speed, acceleration capabilities, and the time and distance limits, a DIST.INCR, a TIME.INCR, and a final SPD are computed (see Figure 14).

Lines 159-165 Sample the terrain using routines ELEVG and MOVE. LIMITS to get a target speed SPD.LIMIT, and limits ACCEL.LIMIT > 0 and DECEL.LIMIT < 0.

Line 166 Computes a distance limit for the move. Note the user supplied MAX.DIST.INCR which forces periodic re-sampling of the terrain.

Lines 169-178 Handle the frequently occurring special case where speed is constant throughout the move increment. Simple manipulation of the movement equation

$$d = v_0 t$$

yields a time increment and a distance increment for the move.

Lines 179-205 Consider the more complex situation where acceleration occurs. The movement equation

$$d = v_0 t + \frac{1}{2} a t^2$$

```
'DIRN.COMP'
IF D.TO.MCP IS LESS THAN ZERO.LEVEL,
    GO TO MCP.REACHED
ELSE
LET DEL.X = X.DEST - X.CURRENT(VEH)
LET DEL.Y = Y.DEST - Y.CURRENT(VEH)
LET DIR.OF.MVMT(VEH) = ARCTAN.F(DEL.Y,DEL.X)
'ANGLES'
LET SALPH = SIN.F(DIR.OF.MVMT(VEH))     LET CALPH = COS.F(DIR.OF.MVMT(VEH))
```

Figure 13. MOVE Routine - Segment ii

```
'NEW.INCR'   CALL ELEVG GIVEN X.CURRENT(VEH), Y.CURRENT(VEH)      YIELDING
     Z.CURRENT(VEH), GRAD.X, GRAD.Y
IF FINAL EQUALS 1, GO TO END.MOVE
ELSE
LET SLOPE = GRAD.X * CALPH + GRAD.Y * SALPH
CALL MOVE.LIMITS GIVEN VEH, SLOPE          YIELDING SPD.LIMIT, ACCEL.LIMIT,
    DECEL.LIMIT
LET DIST.LIMIT = MIN.F(D.TO.MCP, MAX.DIST.INCR)
LET DEL.SPD= SPD.LIMIT - SPD(VEH)
IF ABS.F(DEL.SPD) IS LESS THAN 0.1
''EASY CASE -- NO ACCELERATION --
    LET DIST.INCR = REM.MOVE.TIME * SPD.LIMIT
    IF DIST.INCR IS GREATER THAN DIST.LIMIT,
    ''MOVE STOPPED BY DISTANCE LIMIT
         LET DIST.INCR = DIST.LIMIT
         LET TIME.INCR = DIST.INCR / SPD.LIMIT
    ELSE
    ''MOVE STOPPED BY TIME LIMIT
         LET TIME.INCR = REM.MOVE.TIME
    ALWAYS GO TO MOVE.IT
ELSE  ''HARD CASE -- ACCELERATION REQUIRED --
    IF DEL.SPD IS LESS THAN 0, LET ACCEL.LIMIT = DECEL.LIMIT
    ALWAYS LET TIME.REQ = DEL.SPD / ACCEL.LIMIT
    LET DIST.REQ = SPD(VEH)*TIME.REQ + 0.5 * ACCEL.LIMIT * TIME.REQ **2
    IF TIME.REQ IS GREATER THAN REM.MOVE.TIME,
    ''SPD.LIMIT CANNOTBE ATTAINED IN REMAINING TIME, SO CHANGE LIMIT
         LET SPD.LIMIT = SPD(VEH) + ACCEL.LIMIT * REM.MOVE.TIME
         LET DIST.INCR = SPD(VEH) * REM.MOVE.TIME + 0.5 * ACCEL.LIMIT *
              REM.MOVE.TIME ** 2
    ELSE  ''SPD.LIMIT CAN BE ATTAINED
         LET DIST.INCR = DIST.REQ + (REM.MOVE.TIME - TIME.REQ)*SPD.LIMIT
    ALWAYS
    IF DIST.INCR IS LESS THAN DIST.LIMIT,
    ''MOVE WILL BE STOPPED BY TIME LIMIT
         LET TIME.INCR = REM.MOVE.TIME
         LET SPD(VEH) = SPD.LIMIT
    ELSE   ''MOVE STOPPED BY DISTANCE LIMIT
          LET DIST.INCR = DIST.LIMIT
         IF DIST.LIMIT IS LESS THAN DIST.REQ,
              LET TIME.INCR = (SQRT.F(SPD(VEH)**2+2.*ACCEL.LIMIT*DIST.LIMIT)
                   -SPD(VEH))/ACCEL.LIMIT
              ADD TIME.INCR * ACCEL.LIMIT TO SPD(VEH)
         ELSE
              LET TIME.INCR = TIME.REQ +(DIST.LIMIT-DIST.REQ)/SPD.LIMIT
              LET SPD(VEH)=SPD.LIMIT
         ALWAYS
    ALWAYS
```

Figure 14. MOVE Routine, Segment iii

is manipulated in various ways to again yield a time and a distance increment and also the final speed of the vehicle at the end of the move increment.

In each case, the results of segment iii), TIME.INCR, DIST.INCR, and SPD(VEH) are passed on to segment iv) to actually perform the movement increment.

<u>Segment iv)</u> Update location and time

Segment iv) performs the move by changing the vehicle's X and Y coordinates. The code which follows is self explanatory.

```
206    'MOVE.IT'     SUBTRACT TIME.INCR FROM REM.MOVE.TIME
207    ADD DIST.INCR * CALPH TO X.CURRENT(VEH)
208    ADD DIST.INCR * SALPH TO Y.CURRENT(VEH)
209    SUBTRACT DIST.INCR FROM D.TO.MCP
```

<u>Segment v)</u> Check whether finished (see Figure 15 for Code).

Various occurrences can end a MOVE call. If the move time has expired, then except for updating the elevation we are finished (line 210).

If D.TO.MCP has been exceeded, then a new direction computation is needed, and, if the MCP reached is a real one, we want to aim toward the next MCP (lines 213-233).

If a mine detonation or a minefield entry or exit has occurred, we must pause to assess the implications (damage, lower mine plow, ...). Coding for these functions is not yet written, but the tests for them will be included at the beginning of this segment.

At the end of the move time specified for this move, the T.SPD(VEH) is updated to the current simulation time and control returns to the calling program. (lines 235-236).

```
IF REM.MOVE.TIME IS LESS THAN   0.01      LET FINAL = 1
REGARDLESS
IF D.TO.MCP IS LESS THAN ZERO.LEVEL,
'MCP.REACHED'
    IF FAKE.MCP EQUALS 1
         LET FAKE.MCP = 0
         GO TO NEW.MCP
    ELSE
    IF MCP = 0
         LET FINAL = 1
         GO TO NEW.MCP
    ELSE
    IF DIR.ON.RT(VEH) EQUALS 0    ''MCP NUMBERS INCREASE
         IF NEXT.MCP(VEH) EQUALS DIM.F(ROUTE.DATA(ROUTE(VEH),*))/3
              LET NEXT.MCP(VEH) = 0
              GO TO NEW.MCP
         ELSE
              ADD 1 TO NEXT.MCP(VEH)      GO TO NEW.MCP
    ELSE      ''MCP NUMBERS DECREASE
         IF NEXT.MCP(VEH) EQUALS 1
              LET NEXT.MCP(VEH)=0
              GO TO NEW.MCP
         ELSE
              SUBTRACT 1 FROM NEXT.MCP(VEH)     GO TO NEW.MCP
ELSE  GO TO NEW.INCR
'END.MOVE'    LET T.SPD(VEH) = TIME.V
RETURN
END
```

Figure 15. MOVE Routine, Segment v.

IV. INTERFACE

This section of the report will concentrate on the data flow between the MOVE routine and the rest of the STAR model, showing how to use the MOVE routine in each of the possible modes.

A. MODES OF USE

1. The most frequently used mode will be movement from one position area to another. To initiate this (combination) mode the STAR model should set AREA.START and AREA.END to the appropriate area numbers, set MV.STATE = 1, set T.SPD to the time at which the move was to have started, and call MOVE(VEH). The MOVE routine will select the route, start the vehicle moving, and return with a MVSTATE of 2. Subsequent calls to MOVE to continue the movement should leave MV.STATE= 2. When the vehicle reaches its final position in AREA.END, the MOVE model will return MV.STATE = 4.

If the user wants the vehicle to select the best position in AREA.END, POS.IN.PLT.AREA(VEH) should be set to 0 prior to the first call to MOVE and left unchanged thereafter.

Movement will be in the formation specified by the vehicle (or forced by the terrain) unless FORM.CODE = 0 in which case movement will be along the route.

Prior to some subsequent call to move; (while MV.STATE still = 2), if AREA.START and AREA.END are interchanged, then the MOVE model will reverse the vehicle's direction of movement along the route.

2. A simple mode of movement is straight line motion from current location to a position area (where current location may also be in a position area).

To achieve this, the user should set

MV.STATE(VEH) = 2 (no route select)

ROUTE(VEH) = any number except 0

(the route will not actually be used)

NEXT.MCP(VEH) = 0

AREA.END(VEH) = the desired area

T.SPD (VEH) = the time at which the move was to have started

POS.IN.PLT.AREA(VEH) = the desired position # (or 0 for best position) and call MOVE(VEH).

The straight line movement will continue with subsequent MOVE calls until the vehicle reaches the position (signalled by MV.STATE = 4 on return). Formations cannot be used in this mode as there are no routes.

3. The simplest mode of <u>movement is movement in a specified direction</u>. To achieve this the user should set

DIR.OF.MVMT(VEH) = the desired direction

MV.STATE(VEH) = 2

ROUTE(VEH) = 0

and call MOVE(VEH). Movement in the specified direction will continue, upon subsequent calls to move, with no consideration given to routes, position areas, or formations. <u>CAUTION</u>: this mode may lead to vehicles driving off the map with unpredictable (unusually disastrous) results in the simulation. The user can stop the movement by setting MV.STATE = 0, 3, 4, or 5 at any time.

4. <u>Stopping</u>. In any of the above modes of movement, if MOVE is called with MV.STATE = 3, then the MOVE.LIMITS routine should set SPD.LIMIT = 0 and the vehicle will try to stop. If the deceleration rate allows, the vehicle will stop with SPD(VEH) = 0 on return. If the movement time is too short to allow stopping, the vehicle will decelerate as much as possible, and upon subsequent MOVE calls, will stop. To start again, set MV.STATE = 2 and call MOVE.

B. Current Use in STAR

In the current version of the STAR model, the decisions of when and where to initiate movement are handled by the movement decision logic and movement coordination logic which are documented in reference [4]. Mode 1) above is used exclusively. Whenever the model needs to know the location of a vehicle, the MOVE routine is then called to update its position. This is done periodically for all vehicles in event STEP.TIME and also at other times for individual vehicles involved in a firing event as either shooter or target.

V. EXTENSIONS

The primary aspects of movement in the STAR model which are not covered in this report are

a) deciding when and where to move (see ref. [4].)

b) movement of aircraft (see ref. [3].)

c) interaction of moving vehicles with minefields and obstacles of various kinds.

A general field structure has been developed to include minefields, obstacles, and other battlefield features. This module is currently undergoing tests and will be the subject of a future report.

REFERENCES

[1] HAGEWOOD, E.G. and WALLACE, W.S., Simulation of Tactical Alternative Responses (STAR), M.S. Thesis, Naval Postgraduate School, Dec. 1978.

[2] HARTMAN, J.K., "Parametric Terrain and Line of Sight Modelling in the STAR Combat Model", Naval Postgraduate School, Technical Report NPS55-79-018, August 1979.

[3] CALDWELL, W.J. and MEIERS, W.D., An Air to Ground and Ground to Air Combined Arms Combat Simulation (STAR-AIR), M.S. Thesis, Naval Postgraduate School, September 1979.

[4] PARRY, S.H. and KELLEHER, Jr., E.P., "Tactical Parameters and Input Requirements for the Ground Component of the STAR Combat Model, NPS55-79-023, October 1979.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
4.	Professor James K. Hartman Code 55Hh Department of Operations Research Naval Postgraduate School Monterey, California 93940	40
5.	Professor S. H. Parry, Code 55Py Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
6.	LTC Edward P. Kelleher, Code 55Ka Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
7.	Professor Arthur L. Schoenstadt, Code 53Zh Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
8.	Office of the Commanding General U.S. Army TRADOC Attn: General Donn A. Starry Ft. Monroe, Virginia 23651	1
9.	Headquarters U.S. Army Training & Doctrine Command Attn: ATCG-T (Col. Ed Scribner) Ft. Monroe, Virginia 23651	1

10. Headquarters | 1
U.S. Army Training & Doctrine Command
Attn: Director, Analysis Directorate
Combat Developments (MAJ Chris Needels)
Ft. Monroe, Virginia 23651

11. Headquarters | 1
U.S. Army Training & Doctrine Command
Attn: Director, Maneuver Directorate
Combat Developments (Col. Fred Franks)
Ft. Monroe, Virginia 23651

12. Mr. David Hardison | 1
Deputy Under Secretary of the Army
(Operations Research)
Department of the Army, The Pentagon
Washington, D.C. 20310

13. LTG William Richardson | 1
Commanding General
U.S. Army Combined Arms Center
Ft. Leavenworth, Kansas 66027

14. Director | 1
Combined Arms Combat Development Activity
Attn: Col. Reed
Ft. Leavenworth, Kansas 66027

15. Director, BSSD | 1
Combined Arms Training Development Activity
Attn: ATZLCA-DS
Ft. Leavenworth, Kansas 66027

16. Director | 1
Combat Analysis Office
Attn: Mr. Kent Pickett
U.S. Army Combined Arms Center
Ft. Leavenworth, Kansas 66027

17. Command and General Staff College | 1
Attn: Education Advisor
Room 123, Bell Hall
Ft. Leavenworth, Kansas 66027

18. Dr. Wilbur Payne, Director | 1
U.S. Army TRADOC Systems Analysis Activity
White Sands Missile Range, New Mexico 88002

19. Headquarters, Department of the Army | 1
Office of the Deputy Chief of Staff
for Operations and Plans
Attn: DAMO-2D
Washington, D.C. 20310

20. Commander 1
U.S. Army Concepts Analysis Agency
8120 Woodmont Avenue
Attn: MOCA-SMS (CPT Steve Shupack)
Bethesda, Maryland 20014

21. Commander 1
U.S. Army Concepts Analysis Agency
Attn: LTC Earl Darden-MOCA-WG
8120 Woodmont Avenue
Bethesda, MD. 20014

22. Director 1
U.S. Army Night Vision & Electro-optical Lab.
Attn: DEL-NV-VI-Mr. Bob Hermes
Fort Belvoir, VA 22060

23. Director 1
U.S. Army Material Systems Analysis Activity
Attn: Mr. Will Brooks
Aberdeen Proving Grounds, Maryland 21005

24. Director 1
Combat Developments Experimentation Command
Attn: CPT William J. Caldwell
Fort Ord, California 93941

25. Director 1
Armored Combat Vehicle Technology Program
Attn: Col. Fitzmorris
U.S. Army Armor Center
Ft. Knox, Kentucky 40121

26. Col. Frank Day 1
TRADOC Systems Manager - XM1
U.S. Army Armor Center
Ft. Knox, Kentucky 40121

27. Director 1
Combat Developments, Studies Division
Attn: MAJ W. Scott Wallace
U.S. Army Armor Agency
Fort Knox, KY 40121

28. Commandant 1
U.S. Army Field Artillery School
Attn: ATSF-MBT (CPT Steve Starner)
Fort Sill, Oklahoma 73503

29. Director 1
Combat Developments
Attn: Col. Clark Burnett
U.S. Army Aviation Agency
Fort Rucker, Alabama

30. Director 1
Combat Developments
U.S. Army Infantry Agency
Fort Benning, GA

31. CDR. MICOM 1
ATTN: DRSMI-YC (CPT. HAGEWOOD)
Redstone Arsenal, AL 35809

32. Director 1
Combat Developments
ATTN: CPT William D. Meiers
U.S. Army Air Defense Agency
Fort Bliss, TX 79905

33. Commander 1
U.S. Army Logistics Center
ATTN: ATCL-OS-Mr. Cammeron/CPT Schuessler
Fort Lee, VA 23801

34. Commander, USAMMCS 1
ATTN: ATSK-CD-CS-Mr. Lee/Mr. Marmon
Redstone Arsenal, AL 35809

35. Commander 1
U.S. Army Combined Arms Center
ATTN: ATZL-CA-CAT (R.E. DeKinder, Jr.)
Fort Leavenworth, KA 66027

36. Director 1
U.S. Army AMSAA
ATTN: DRXSY-AA (Mr. Tom Coyle)
Aberdeen Proving Grounds, MD 21005

37. R. Stampfel, Code 55 1
Department of Operations Research
Naval Postgraduate School
Monterey, California 93940

38. Dean of Research 1
Naval Postgraduate School
Monterey, California 93940